Institute of Software Technology
Reliable Software Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Fachstudie

Evaluating Open-source Tool Stacks for Application Performance Diagnostics

Jan Ruthardt
Nico Poier
Thomas Breunig

Course of Study:	Softwaretechnik
Examiner:	Dr.-Ing. André van Hoorn (Prof.-Vertr.)
Supervisor:	Thomas F. Düllmann M.Sc., Teerat Pitakrat, M.Sc.
Commenced:	August 14, 2017
Completed:	February 14, 2018
CR-Classification:	I.7.2

Abstract

Evaluation of different Open-Source Application Performance Management tools and stacks. Testing of the tools takes places at the RSS Infrastructure (Kubernetes Cluster) and a instance of Sockshop (Microservice Webshop). The goal is to set up a Stack of different tools that is best for monitoring the RSS Infrastructure.

Contents

List of Figures

List of Tables

List of Acronyms

FR Fehlerrate

List of Listings

List of Algorithms

Chapter 1

Introduction

Nowerdays its very Common in IT to have Distributed Systems in Location all over the Globe. To be able to provide the best user Experience its important to Monitor these Networks by only few People sitting in one or more Location. Important here is the Availability and Reliability also as the Response Time of the System.

To tackle these kinds of Tasks Application Performance Management Tool were build. They are available in a wide range of Costs and Qualities. They differ a lot in their Architecture and style of tackling Problems. This is why we decided to make a Comparison of some large Open-Source tool stacks available on the Market

Thesis Structure

In the first part the Paper describes the general aspects of Monitoring and the general Metrics. The part also discusses the characteristics of the environments and their special interfaces. After the introduction the tools will be introduced on there own.

Technical Details: In this chapter all technical aspects of the Test environments and Tools will be explained. Also the Term Stack will be illustrated.

Collectors: Explains all details and methods of the client based tools.

Database: Analyzes the different types of Methods of Storing Time related Data

Visualizaion: Illustrates how time related data can be presented in general and which Visual limitation to tackle

Alerting: Here the different Approaches and methods to inform the User/Administrator about any Problems are discussed

InfluxData: Company that was Founded 2012 and Provides a Full Stack Open-Source APM tool.

Elastic: The Elasticsearch BV provides a tool stack of APM tools. In the middle of this stack is the Elasticsearch Application which is a searching engine written in Java.

Prometeus: Prometheus is a Open-Source Tool stack for monitoring and alerting, with focus on reliability and simplicity.

Zabbix: Zabbix is an open source monitoring software for networks and applications for enterprise use

Goals

Goal of the Study is to print out the benefits and disadvantages of the popular Open-Source Tools and Stacks Available on the Market. The work wants to Illustrate the features and technologies of the tools to make it easier for the Reader to get an overview over the different Software approaches. In particular the tools will be tested in their ability to interact with modern Cloud technologies like Docker and Kubernetes. Furthermore they will be compared by their Ability to integrate in existing environments and support of common tools and Interfaces. Moreover the Cross Compatibility of the stacks will be tested to get the best out of the tool pool.a

Chapter 2

Technical Data

2.1. General Stack

To explain a Stack in general, distinguish the difference between monitoring and logging. First, monitoring is running in the background and constantly collects the system data's, these data is collected by specific metrics. However logging is only triggered by a definite event or exception. With this knowledge it is a easier understanding how to establish a monitoring stack. Therefor 4 different types of tools are needed. First at all one for collect data from the system by specific metrics, the collector. Second to store, maintain and querying them, the database and last to visualize the data, the visualization tool. Often an alerting tool is also needed, but this is often integrated in the visualization tool. The logging stack has similar

2.2. Collector

To get Data in a centralized spot a tool is needed to collect the data were its generated and transport it to the Server or provide an Interface for the Server to collect the data. Tools for this Purpose a we call Collectors. Were are Collector for every Monitoring Purpose. Its very common that a Collector provides a general interface like an XML or JSON data or can be adapted to variable Databases to get a wide spectrum of Use-Cases. The monitored metrics is dependent on the environment and the collector also has to use over tools that provides system data to get these type of metrics. In general the data that is collected can be split up in System data and Application data. System data are all physical values like CPU load, Ram and Hard Disc Drive usage.These will be providet by cAdvisor (2.2.1) in the case of Kubernetes. Application data is dependent on the application. In the Case of monitoring Kubernetes normally the number of jobs/pods

or the number of connection per time will be monitored. These and over data will be provided by the Api-server(2.2.2) of Kubernetes.

2.2.1. cAdvisor

Container Advisor is tool for collection,Processing and Exporting Data of Containers. It is native Designed for Docker but can be applied to ever other container. All information about the Container is Accessible over a Rest api that gives back a JSON files with all data. A copy of cAdvisor is Deployed within every Kubernetes Pod, so every APM tool can get the metrics of the system.

2.2.2. Api-Server

Api-Server is a tool that provides a REST interface and is a front end for the hole Kubernetes Cluster. Over the Api-Server a user is able to interact with all Components of the cluster. The Api-Server also collects metrics witch are listed below.

- Aggreation Controler Queue: Used for Parallel Processing as an Middelware
- Registration Controller:

2.3. Database

The Databases for a APM are usual time-series based (2.3.1). As every other database its used to make data persistent and perform request over multiple entries to get new informations about critical values and value changes over time. Databases can offer two types of data providing methods. Most of the time the database provides a well defined interface which normals provides a authentication method to insert data into the database. Every of these Snapshot than gets a timestamp.
The over method is that the database preforms a get operation onto a interface provided by the Collector. This type of data-collection is better for static system or must be

2.3.1. Time-Series-Database

This is a special kind of database developed for saving time series data. This data consists of arrays which are indexed by a time stamp. By the term Time-Series also a time ranges cloud be used (as a primary key). These types of Databases can create, enumerate, update, delete and analyze time-series-data. Often they also allow you to merge multiple time-series together and make one. Like each other database, time-series-databases can also filter the data which is normally ordered ascending by time.

2.4. Visualization

The Visualization Tools are used to display the data stored in the databases in a nice and organized way. This is realized with plain text or by graphs. Graphs have the big advantage to be able to display the data changes over time and can very easily illustrate spikes in the data sets.Furthermore Graphs can present data in more than one way which makes it easier for humans to detect abnormal data spikes.

Usually all this information can be accessed via a web interface as this also gives a nice option for logins and distribution of permissions. This is especially useful when the data is very sensitive. Often these tools also implement easy to use Interfaces for Alerting tools, to set conditions for specific alerts, which can save a lot of time.

2.4.1. Graphs

As previously mentioned, the data we collected from the cluster needs to be written out of the Database and displayed in a nice and readable fashion. Thus most visualization tools use graphs to display the collected data. Using graphs not only makes the data easy to read, but it also adds the option to scale the data to our needs and preferences. This can be very useful when looking for trends in a bigger time range.
It also gives the option of color coding the data, which can be useful to either see dangerous values more quickly, or simply render multiple data streams in one graph to compare them or to see them im comparison too the hole system.

2.4.2. Permission Management

Most Visualization Tools have a web interface in which all the data is displayed. To make sure only authorized people can view the data, these tools usually implement

a few permission management methods. These can be ranging from simple login permissions to viewing permissions of specific data streams. Some tools allow for complete customization of the permission settings, while others offer a set of permission templates. The most popular method of authorization seems to be LDAP, as this can be used for simple and complex permission schemes alike.

LDAP

Written-out Lightweight Directory Access Protocol is a Network-protocol on a client-server basis. LDAP describes the communication between the client and the LDAP Directory. The data-structure of of LDAP is the so called Directory Information Tree which is organized by one suffix(root) and nodes.

2.5. Alerting

To inform the developer about the system, a tool is needed which is able to sends warnings about predefined system states. The alerting tool gets one or more error codes from the controller that is normally implemented into the database or visualization tool. Some tools combine With this codes the altering tool sends a warning or error message to all people involved. Most of the tools can send over multiple platforms. The most common are: E-Mail, SMS, telegram and slack. Often tools over man then the mentioned interface and provide a api to integrate other alerting types.

Often the developers don't want to get just one alert with one message, so some of the alerting tool have the possibility to sort the alerts into groups. With this feature its possible to group cascading events that are triggered by failure. In some cases tools also supply a option to divide all alerts into critical alerts and warnings which can help the user to select the important massages.

Chapter 3

Tools

3.1. Installed Tools

3.1.1. Searchlight (Icinga)

3.2. Failed Tools

In the Process of Developing and Evaluating the APM we Discoverd a bunch of Tools that we were not able to install even that they clamed to be optimized to work on Kubernetes . In this Paragraph all the tools we wanted to include in our Report but doesn't are mentioned with a quick description of the Failure.

3.2.1. Graphite

Graphite is mainly for storing and Graphing data and metrics, but brings also tools that are able to collect these Metrics from the system. By the Developer it self there is no Kubernetes installation Provide but there are diverse approaches by third party members to make it runnable on a Cluster. We have tested the Repository from nanit (https://github.com/nanit/kubernetes-graphite-cluster,11.12.2017) to get Graphite running with StatsD (https://github.com/etsy/statsd.git,11.12.2017) as a metric collection tool. The Repo doesn't provide a yaml file by it self to install all the tools. The instruction leads the user to export some Variables needed for the installation. After that a deploy command is provided that pulls the docker repo and than installs it with kubectl on the Cluster. As we tried to execute this command a fail was thrown, The Node Replicas were empty, so no further commands are executable. As we were not able to install the

tool on multiple Kubernetes Clusters, we installed the tool as on the Webside advertised on a Ubuntu System directly. With this installation of Graphit a time-series Data,an Monitoring and an Alerting tool is included. On the test System the Monitoring System gave values that differs from the Linux intern monitoring in values like Cpu usage or RAM. As a Solution to all this Difficulties we decided to not perform further test on the tool.

3.2.2. Icinga

As we tried to install Icinga as we found out that there was no direct support from Incinga for Kubernetes. As our Study only describes the actual state without trying to add somthing we decidet not to try an compile Icinga into Kuberntes on oure own. Later we found out there is a third party Software called searchlight 3.1.1. Its provided by appscode on github.

Chapter 4

Conclusion

Hier bitte einen kurzen Durchgang durch die Arbeit.

Future Work

...und anschließend einen Ausblick

Appendix A

LaTeX-Tipps

A.1. File-Encoding und Unterstützung von Umlauten

Die Vorlage wurde 2010 auf UTF-8 umgestellt. Alle neueren Editoren sollten damit keine Schwierigkeiten haben.

A.2. Zitate

Referenzen werden mittels `\cite[key]` gesetzt. Beispiel: [**WSPA**] oder mit Autorenangabe: **WSPA**.

Der folgende Satz demonstriert 1. die Großschreibung von Autorennamen am Satzanfang, 2. die richtige Zitation unter Verwendung von Autorennamen und der Referenz, 3. dass die Autorennamen ein Hyperlink auf das Literaturverzeichnis sind sowie 4. dass in dem Literaturverzeichnis der Namenspräfix "van der" von "Wil M. P. van der Aalst" steht. **RVvdA2016** präsentieren eine Studie über die Effektivität von Workflow-Management-Systemen.

Der folgende Satz demonstriert, dass man mittels `label` in einem Bibliopgrahie"=Eintrag den Textteil des generierten Labels überschreiben kann, aber das Jahr und die Eindeutigkeit noch von biber generiert wird. Die Apache ODE Engine [**ApacheODE**] ist eine Workflow-Maschine, die BPEL-Prozesse zuverlässig ausführt.

Wörter am besten mittels `\enquote{...}` "einschließen", dann werden die richtigen Anführungszeichen verwendet.

Beim Erstellen der Bibtex-Datei wird empfohlen darauf zu achten, dass die DOI aufgeführt wird.

Listing A.1 lstlisting in einer Listings-Umgebung, damit das Listing durch Balken abgetrennt ist

```
<listing name="second sample">
  <content>not interesting</content>
</listing>
```

A.3. Mathematische Formeln

Mathematische Formeln kann man so setzen. `symbols-a4.pdf` (zu finden auf http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf) enthält eine Liste der unter LaTeX direkt verfügbaren Symbole. Z. B. $\mathbb{N}$ für die Menge der natürlichen Zahlen. Für eine vollständige Dokumentation für mathematischen Formelsatz sollte die Dokumentation zu `amsmath`, ftp://ftp.ams.org/pub/tex/doc/amsmath/ gelesen werden.

Folgende Gleichung erhält keine Nummer, da `\equation*` verwendet wurde.

$$x = y$$

Die Gleichung A.1 erhält eine Nummer:

$$x = y \tag{A.1}$$

Eine ausführliche Anleitung zum Mathematikmodus von LaTeX findet sich in http://www.ctan.org/tex-archive/help/Catalogue/entries/voss-mathmode.html.

A.4. Quellcode

Listing A.1 zeigt, wie man Programmlistings einbindet. Mittels `\lstinputlisting` kann man den Inhalt direkt aus Dateien lesen.

Quellcode im `<listing />` ist auch möglich.

A.5. Abbildungen

Die Figure A.1 und A.2 sind für das Verständnis dieses Dokuments wichtig. Im Anhang zeigt Figure A.4 on page 19 erneut die komplette Choreographie.

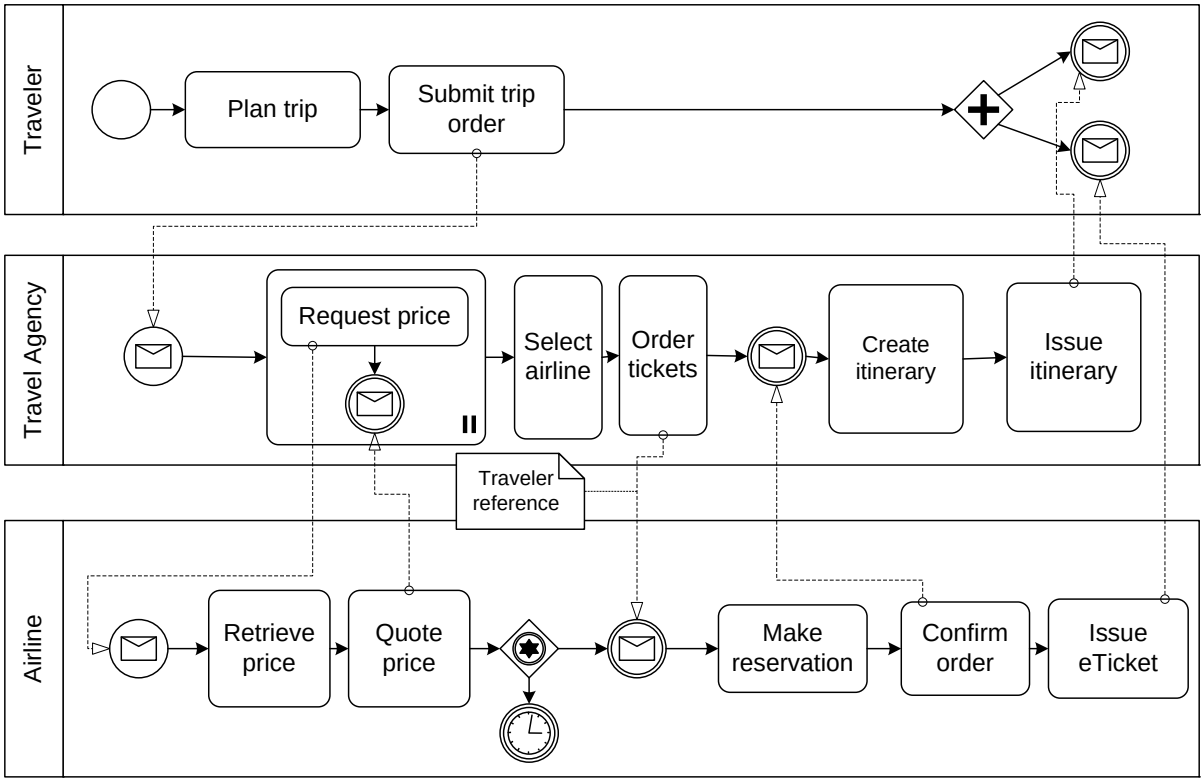


Figure A.1.: Beispiel-Choreographie

Das SVG in **??** ist direkt eingebunden, während der Text im SVG in **??** mittels pdflatex gesetzt ist.

Falls man die Graphiken sehen möchte, muss inkscape im PATH sein und im Tex-Quelltext `\iffalse` und `\iftrue` auskommentiert sein.

A.6. Tabellen

Table A.1 zeigt Ergebnisse und die Table A.1 zeigt wie numerische Daten in einer Tabelle representiert werden können.

A.7. Pseudocode

Algorithm A.1 zeigt einen Beispielalgorithmus.

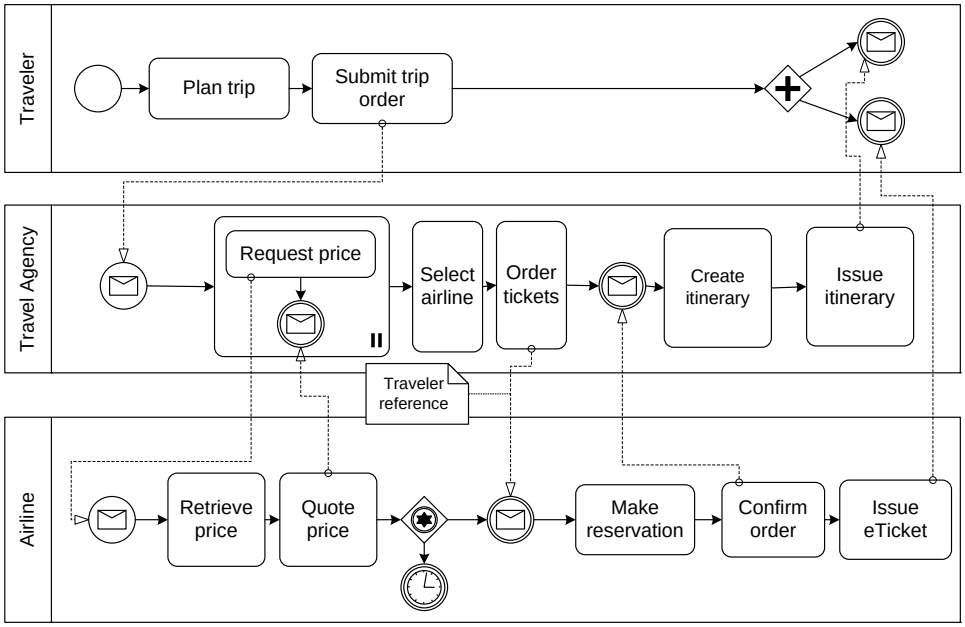
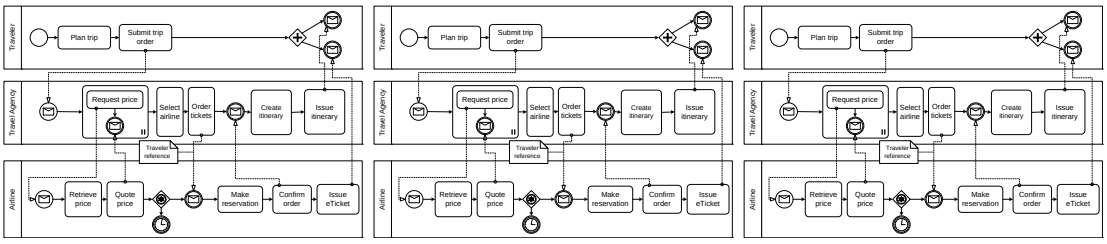


Figure A.2.: Die Beispiel-Choreographie. Nun etwas kleiner, damit `\textwidth` demonstriert wird. Und auch die Verwendung von alternativen Bildunterschriften für das Verzeichnis der Abbildungen. Letzteres ist allerdings nur Bedingt zu empfehlen, denn wer liest schon so viel Text unter einem Bild? Oder ist es einfach nur Stilsache?



(a) **(b)** **(c)** Subcaption if needed

Figure A.3.: Beispiel um 3 Abbildung nebeneinader zu stellen nur jedes einzeln referenzieren zu können. Abbildung A.3b ist die mittlere Abbildung.

zusammengefasst		Titel
Tabelle	wie	in
tabsatz.pdf	empfohlen	gesetzt
Beispiel	ein schönes Beispiel für die Verwendung von “multirow”	

Table A.1.: Beispieltabelle – siehe http://www.ctan.org/tex-archive/info/german/tabsatz/

	Parameter 1		Parameter 2		Parameter 3		Parameter 4	
Bedingungen	**M**	**SD**	**M**	**SD**	**M**	**SD**	**M**	**SD**
W	1.1	5.55	6.66	.01				
X	22.22	0.0	77.5	.1				
Y	333.3	.1	11.11	.05				
Z	4444.44	77.77	14.06	.3				

Table A.2.: Beispieltabelle für 4 Bedingungen (W-Z) mit jeweils 4 Parameters mit (M und SD). Hinweiß: immer die selbe anzahl an Nachkommastellen angeben.

Algorithmus A.1 Sample algorithm

procedure SAMPLE(a,v_e)
  parentHandled $\leftarrow (a = \mathsf{process}) \vee \mathsf{visited}(a'), (a', c, a) \in \mathsf{HR}$
    // $(a', c'a) \in \mathsf{HR}$ denotes that a' is the parent of a
  if parentHandled $\wedge (\mathcal{L}_{in}(a) = \emptyset \vee \forall l \in \mathcal{L}_{in}(a) : \mathsf{visited}(l))$ **then**
    $\mathsf{visited}(a) \leftarrow$ true
    $$\mathsf{writes}_\circ(a, v_e) \leftarrow \begin{cases} \mathsf{joinLinks}(a, v_e) & |\mathcal{L}_{in}(a)| > 0 \\ \mathsf{writes}_\circ(p, v_e) & \exists p : (p, c, a) \in \mathsf{HR} \\ (\emptyset, \emptyset, \emptyset, false) & \text{otherwise} \end{cases}$$
    if $a \in \mathcal{A}_{basic}$ **then**
      HANDLEBASICACTIVITY(a,v_e)
    else if $a \in \mathcal{A}_{flow}$ **then**
      HANDLEFLOW(a,v_e)
    else if $a = \mathsf{process}$ **then** // Directly handle the contained activity
      HANDLEACTIVITY(a',v_e), $(a, \perp, a') \in \mathsf{HR}$
      $\mathsf{writes}_\bullet(a) \leftarrow \mathsf{writes}_\bullet(a')$
    end if
    for all $l \in \mathcal{L}_{out}(a)$ **do**
      HANDLELINK(l,v_e)
    end for
  end if
end procedure

Und wer einen Algorithmus schreiben möchte, der über mehrere Seiten geht, der kann das nur mit folgendem **üblen** Hack tun:

Algorithmus A.2 Description

code goes here
test2

A.8. Abkürzungen

Beim ersten Durchlaf betrug die Fehlerrate (FR) 5. Beim zweiten Durchlauf war die FR 3.

Mit `\ac{...}` können Abkürungen eingebaut werden, beim ersten aufrufen wird die lange Form eingesetzt. Beim wiederholten Verwenden von `\ac{...}` wird automatisch die kurz Form angezeigt. Außerdem wird die Abkürzung automatisch in die Abkürzungsliste eingefügt.

Definiert werden Abkürzungen in der Datei *ausarbeitung.tex* im Abschnitt '%%% acro' mithilfe von `\DeclareAcronym{...}{...}`.

Mehr infos unter: http://mirror.hmc.edu/ctan/macros/latex/contrib/acro/acro_en.pdf

A.9. Verweise

Für weit entfernte Abschnitte ist "varioref" zu empfehlen: "Siehe Appendix A.3 on page 12". Das Kommando `\vref` funktioniert ähnlich wie `\cref` mit dem Unterschied, dass zusätzlich ein Verweis auf die Seite hinzugefügt wird. `vref`: "Appendix A.1 on page 11", `cref`: "Appendix A.1", `ref`: "A.1".

Falls "varioref" Schwierigkeiten macht, dann kann man stattdessen "cref" verwenden. Dies erzeugt auch das Wort "Abschnitt" automatisch: Appendix A.3. Das geht auch für Abbildungen usw. Im Englischen bitte `\Cref{...}` (mit großen "C" am Anfang) verwenden.

A.10. Definitionen

Definition A.10.1 (Title)
Definition Text

Definition A.10.1 zeigt ...

A.11. Verschiedenes

KAPITÄLCHEN werden schön gesperrt...

I. Man kann auch die Nummerierung dank paralist kompakt halten
II. und auf eine andere Nummerierung umstellen

A.12. Weitere Illustrationen

Abbildungen A.4 und A.5 zeigen zwei Choreographien, die den Sachverhalt weiter erläutern sollen. Die zweite Abbildung ist um 90 Grad gedreht, um das Paket `rotating` zu demonstrieren.

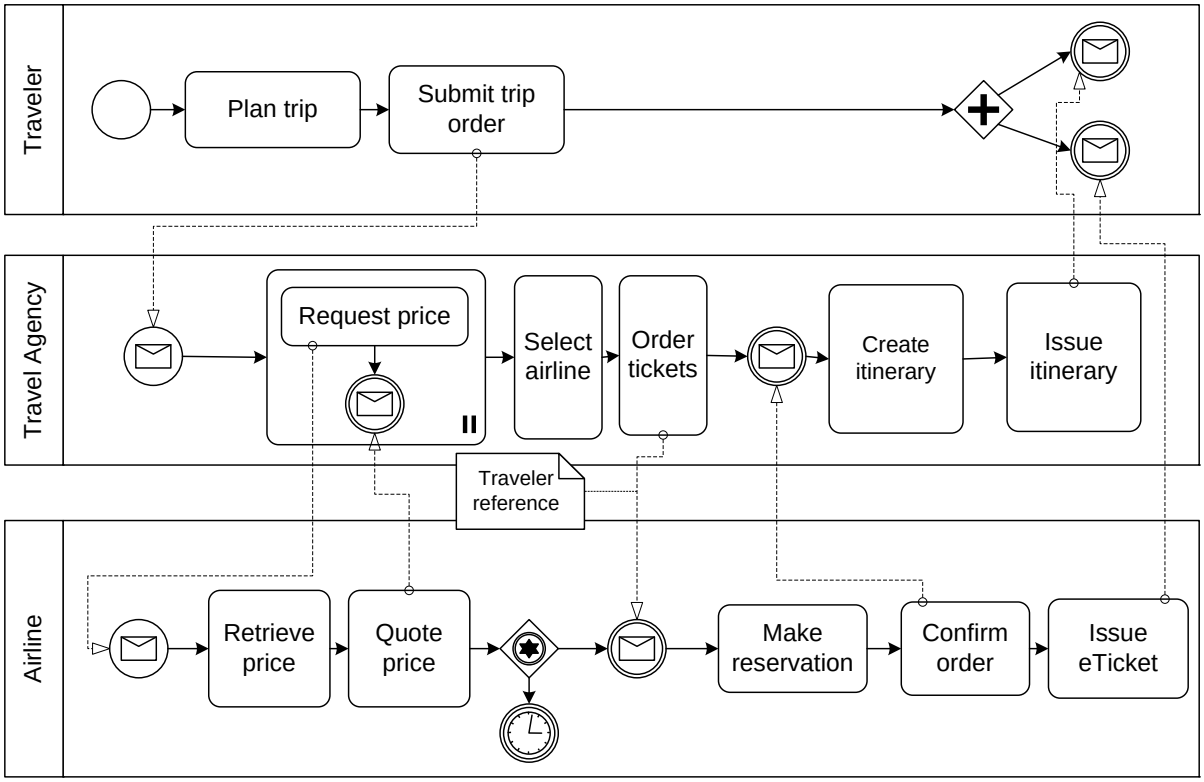


Figure A.4.: Beispiel-Choreographie I

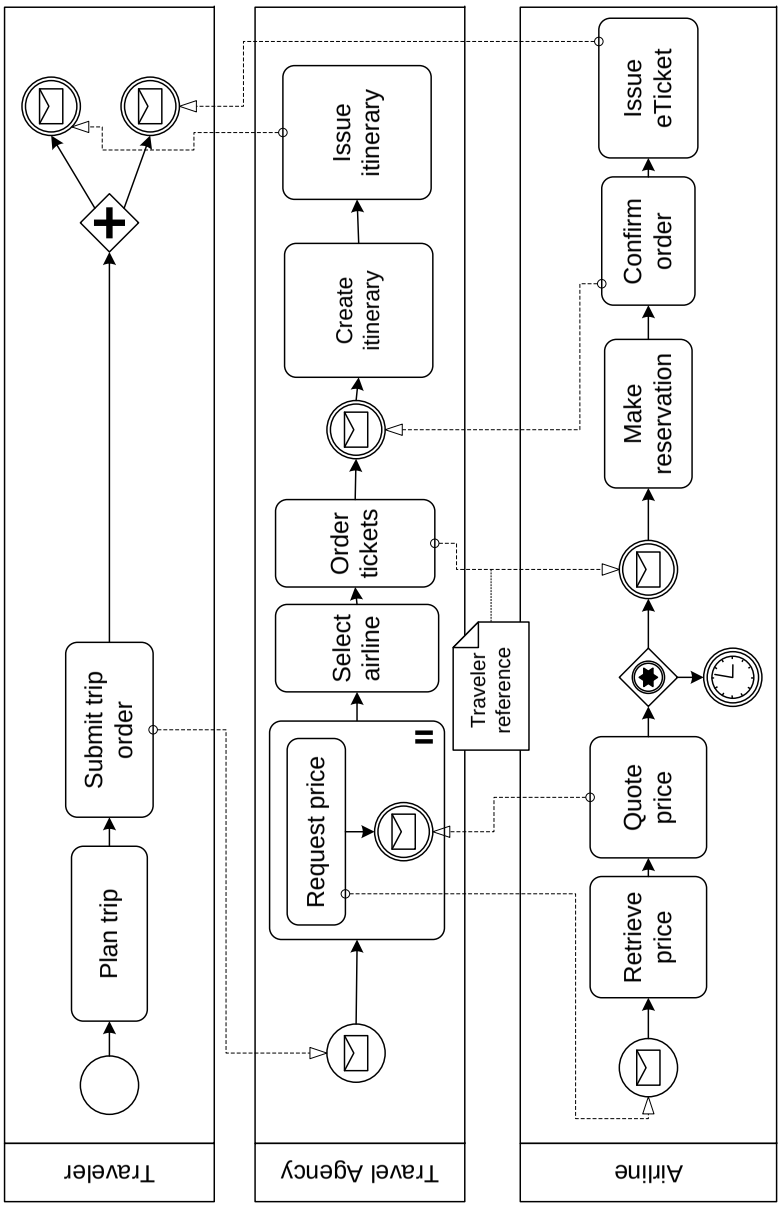


Figure A.5.: Beispiel-Choreographie II

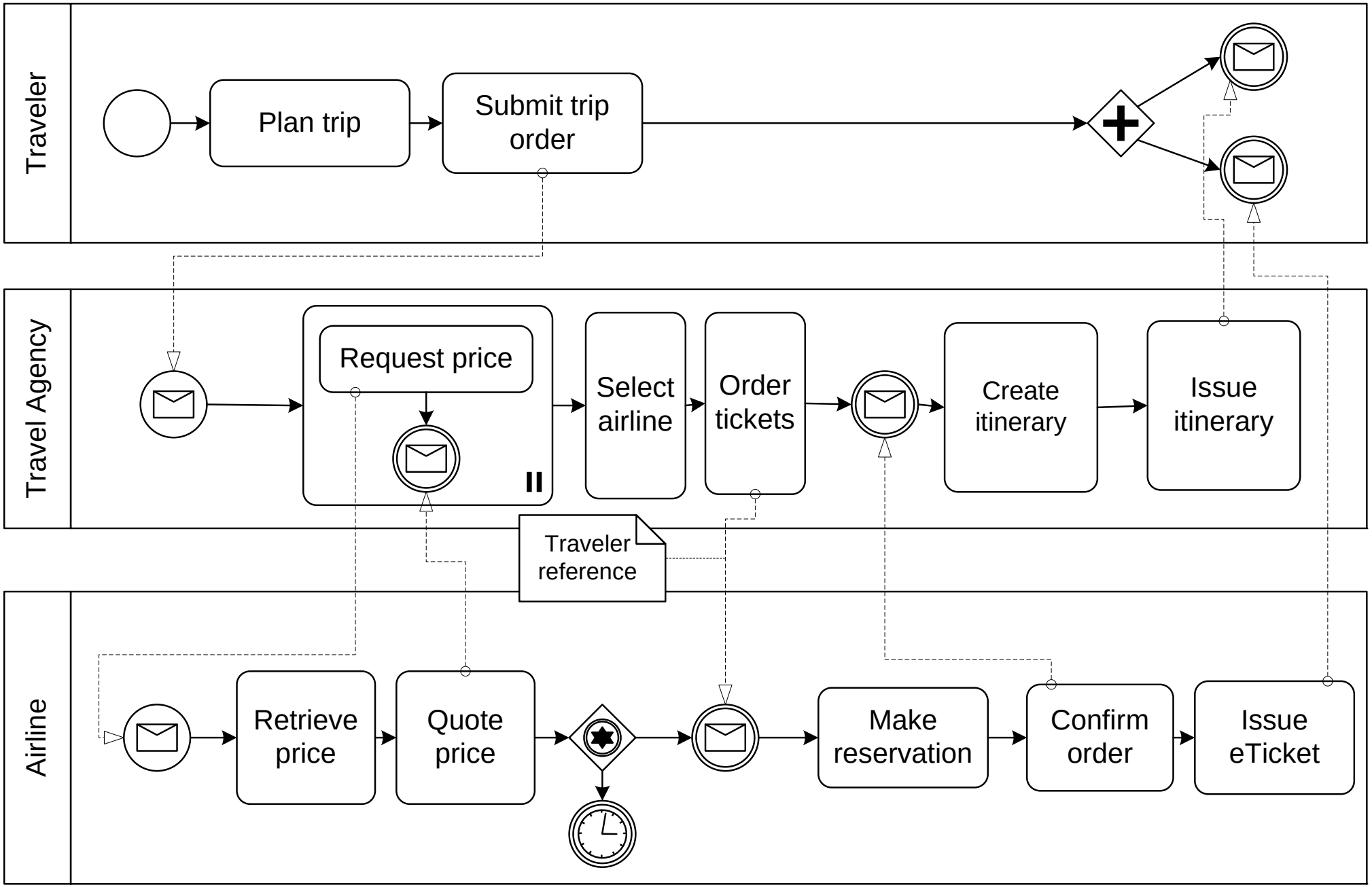


Figure A.6.: Beispiel-Choreographie, auf einer weißen Seite gezeigt wird und über die definierten Seitenränder herausragt

A.13. Schlusswort

Verbesserungsvorschläge für diese Vorlage sind immer willkommen. Bitte bei github ein Ticket eintragen (https://github.com/latextemplates/uni-stuttgart-computer-science-template/issues).

All links were last followed on March 17, 2008.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature